POWER, PERFORMANCE AND AREA EXPLORATION OF BLOCK MATCHING ALGORITHMS MAPPED ON PROGRAMMABLE PROCESSORS

N. Kroupis, M. Dasigenis, A. Argyriou, K. Tatas, D. Soudris, A. Thanailakis

VLSI Design and Testing Center,
ECE Dept., Democritus University of Thrace,
67 100 Xanthi, Greece

N. Zervas, C. E. Goutis

VLSI Design Lab
ECE Dept., University of Patras,
26 500, Patras, Greece

ABSTRACT

A comparison study of four blocking matching algorithms in terms of performance, energy consumption, and area, assuming five matching criteria is presented. Two widely-used programmable processor cores, namely the embedded processor ARM with specialized memory hierarchy and the general purpose processor, Pentium© are chosen for performing the comparative study, and eventually, determining the most efficient solution for certain design specifications. Adopting a systematic design methodology, efficient high-level transformations are applied into the considered algorithm structures resulting into performance-, energy-, and area-optimized solutions. The plethora of alternative implementations under the different distance criteria and processor cores, offers the flexibility to the designer finding the most suitable for him/her solution, given the design specifications.

1. INTRODUCTION

The number of multimedia systems used for exchanging information is rapidly increasing nowadays. Portable multimedia applications, such as video phones, multimedia terminals and video cameras, are available. Such applications that make use of large multi-dimensional array-type data structures, require a very large amount of memory. Although the cost of memory as well as the memory size continuously decreasing, the constant increase in these data dominant applications make the memory cost most of the times, one of the vast contributions to the total system cost.

A very important task in the process of designing a multimedia system is the reduction of the total number of transfers between the processor core and the background (off-chip) memory. This can be accomplished by performing high level transformations on the algorithm structure. It has been proved that the high level transformations are the most promising ones for large gains in power consumption [1]. A systematic methodology for the reduction of memory power consumption has been proposed in [2]. This methodology includes the application of global, loop and data flow transformations.

There are two approaches for the implementation of block matching algorithms. The first is to use custom hardware [3], [4] and [5]. This approach may be power and area efficient but it lacks of flexibility, since only a specific algorithm can be executed [8]. The second solution is to use programmable processors, which is less efficient for power and area than the previous approach but it allow us to implement multiple algorithms on the same target architecture platform [6] and [7]. Thus, it implements a broader class of applications, within reduced design cycle. In other words, such processor platforms allows to the designer meeting faster the tight time-to-market constraints. The problem of exploring the design space for multimedia applications executed in embedded processors is relatively new. More specifically a systematic methodology for the reduction of memory power consumption was presented in [8]. However, the proposed methodology targets only custom hardware architectures and did not tackle with the problem when programmable processors are used.

In this paper power, performance and area exploration results of four well known motion estimation (ME) kernels, being executed in two different processor architectures, under five different matching criteria, are presented. The way in which a series of transformations affect power and performance is analyzed. The aim of the research under consideration is to analyze the effect of transformations into different processor architectures, namely a Reduced Instruction Set Computer (RISC) using the ARM 7 processor core, which is an embedded programmable core, and a Complex Instruction Set Computer (CISC) using the Pentium© II processor, which is a general purpose processor.An application specific memory hierarchy was used in the case of the embedded ARM processor, while a fixed memory hierarchy was used for the Pentium© processor.

2. BASIC BACKGROUND: BLOCK MATCHING ALGORITHMS

Our demonstrator applications was selected to be four well known motion estimation algorithms: *i*) Full Search, *ii*) Hierarchical Search, *iii*) Parallel Hierarchical One Dimensional Search, and *iv*) 3-Step Logarithmic Search [9]. These algorithm were chosen because they are used in a great number of image and video processing applications. Our experiments were carried out using the luminance components of QCIF frame (144x176) format. Reference window was selected to include 15x15 candidate blocks, while blocks of 16x16 pixels were considered.

2.1. Search Area Strategies

2.1.1. Full Search

Full Search is the principal motion estimation algorithm that searches all the blocks of the current frame for similarities with the previ-

This work was supported by the project ED 501 PENED '99 funded by G.S.R.T of Greek Ministry of Development and European Union.

0-7803-6725-1/01/$10.00 ©2001 IEEE

ous frame and guarantees to find the best match. This is the simpler method to find the motion vector for each block and its computational complexity is 513,216 MOPS (Mega Operations Per Second) for frame size 176x144 pixels, and search space $p = 7$[9]. The algorithm structure has three double nested loops. A block of the current frame (outer loop) is compared to a number of candidate blocks (middle loop). In the inner loop, a distortion criterion is computed to perform the comparison.

2.1.2. *Hierarchical Search*

A motion estimation algorithm that use a combination of both fewer search locations and fewer pixels in computing the matching block. In this algorithm, two low-resolution versions of the current picture and the reference picture are formed by subsampling both of them by a factor of two and four. The motion vector search begins at the lowest resolution picture (level 2). The search space is one fourth of the original one. At level one the motion vector search is performed with the origin being the block that corresponds to the level 2 block where there is a closest match and a search region of [-1,1] pixel around it. At level zero the search origin is located at the block which corresponds to the level one block where the corresponding distance criterion is minimized and the search region is again [-1,1] pixels around the block. The location that yields the smallest distance criterion corresponds to the final motion vector output. A Full-Search method is used for motion estimation at each level of the hierarchy, therefore the algorithm structure is very similar. The computational complexity of this algorithm is 29,224 MOPS[9].

2.1.3. *Parallel Hierarchical One-Dimensional Search*

In this algorithm the search occurs on the two dimensions of the search space independently, and getting the motion vector by the locations (dx, dy) of the best matching block on X and on Y dimension. The search begin with step $S = 2^{\lfloor \log_2 p \rfloor}$, and in every cycle we compare the three blocks on X that are located $(x-S, y)$, (x, y) and $(x+S, y)$ and three on Y dimension $(x, y-S)$, (x, y) and $(x, y+S)$. In the second cycle the pixel $(x+dx, y+dy)$ is set to be the new starting point and set $S = S/2$ as the new search step around the point with the best matching of the last cycle. Finally, the motion vector is obtained when the step becomes zero. Its computational complexity is 29,652 MOPS[9].

2.1.4. *3-Step Logarithmic Search*

Three step logarithmic is very similar to PHOD search. In the first step, the $[-p, p]$ search rectangle is divided into two areas: one inside a $[-p/2, p/2]$ rectangle and one outside it. Instead of searching the whole $[-p/2, p/2]$ area, the distance criterion is computed at the origin and eight points at the perimeter. Using the best match among the eight points as a new origin, we perform a new similar search. Then, we use the new best match as the origin and perform a third search among eight new perimeter points. This algorithm requires 57,024 MOPS[9].

2.2. Distance Criteria

Block-matching motion estimation algorithms gain the motion vector by minimizing a cost function, which is generally based on a distance criterion. Various distance criteria have been proposed

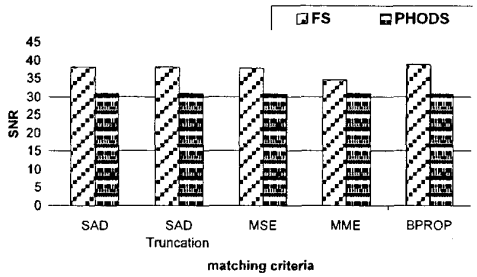

Fig. 1. SNR measurements for different distance criteria, for two motion estimation algorithms

and analyzed in literature [10]. All of them vary in terms of the implementation complexity and efficiency, concerning the approach used to find the global optimum. In our measurements we used 5 distance criteria: i) Mean Square Error, MSE, which produces superior results, as the MSE can be interpreted as Euclidean distance between two matching-blocks, ii) Sum of Absolute Differences, SAD, which calculates the difference of every pixel of the matching block, iii) SAD summation truncation, which is similar to the SAD, with the difference of the break in stopping the SAD calculation, when the current interim SAD value is higher then the minimum SAD calculated so far, iv) Minimized Maximum Error, MME which is similar to the SAD but only an 8-bit comparator unit is required for the comparison, v) Binary Level Matchingg Criterion, BPROP which simply counts the number of matching pixels. Finally, measurements concerning the SNR of the same algorithm using different motion estimation criteria, show that the SNR remains unchangeable (Fig. 1). These measurements were obtained using a typical video sequence of two hundrend frames. We selected five distance criteria, in order to examine the impact of our methodology when the SNR requirements are flexible. This concludes, that the designer can select the distance criterion by examining the computational complexity of it, and the desired picture quality in the other. We also selected these matching criteria that can offer advantages when implemented in programmable processors.

3. OPTIMAZATION METHODOLOGY

The fact that in multimedia applications the power related to memory transfers is the dominant factor in total power cost, motivate us to find an efficient method to reduce them. For that purpose we create a data memory hierarchy that exploits the temporal locality of the data, by reusing them. If there is a sufficient reuse of the data, it can be advantageous to copy the data that are used frequently to a smaller memory, such that for the following usages, a data element can be read from the smaller memory instead of the larger memory. The smaller memories require less power per access and as a result they decrease the total power consumption. However, we must first determine the certain data sets, which are heavily re-used in a short period of time and therefore are appropriate to be placed in a separate memory. For that purpose, we performed an exhaustive data reuse exploration of the application's data over time [2]. The main features of the used methodology and exploration results, are also reported in [2], assuming only one kind of processor (i.e. RISC) and one criterion (i.e. SAD). Here, extension

of the exploration-optimization methodology to more distance criteria is presented.

Since our target architecture consists of programmable processors, we should take into consideration the power dissipation due to instruction memory as well. Previous work [2] has revealed that this power parameter is a significant part of total system's power, and thus, it should not be ignored. Also, it depends on both number of executed instructions and the size of the application code. Particularly, the number of executed instructions determines how many times the instruction memory is accessed, while the code size determines the memory size. The cost function used for our data reuse exploration on all target architectures is evaluated in terms of power, performance, and area, taking into account both data and instruction memories.

3.1. High Level Estimation Model

High level description of an application, e.g. algorithm-level, the design trajectory from the higher to lower design levels imposes making accurate estimation of certain design parameters such as power, area, and performance. In other words, we need a high-level estimation model to characterize an application through the calculation of critical factors. The model used is discussed in detail in [2]. Here, we briefly name the used models. In order to estimate: *i*) the power dissipation we used Landman's model [11], *ii*) the memory area we used Mulder's model [12], *iii*) the performance of ARM processor we used ARMulator Tool, and *iv*) the performance of Pentium© processor we used Vtune.

4. EXPERIMENTAL RESULTS

In this section we present the results after our exploration optimization methodology has been applied. We used as our target vehicles the four aforementioned Motion Estimation algorithms. Additionally, we used five different matching criteria in order to explore the impact of our methodology when the SNR requirements are flexible. In this way we can incorporate in our algorithms less computationally complex matching criteria and obtain solutions that are faster and more power efficient. Due to the huge number of data-reuse transformations, we selected to present, only the original motion estimation algorithm (brighter bar) and at the next bar the best data-reuse transformation that satisfies the power-delay criterio (darker bar).

4.1. Power dissipation

Fig. 2 - 5 illustrate the total memory power consumption for the four ME algorithms. The first important observation that can be deduced from the results, is that the total power consumption decreases after the transformations have been applied. Furthermore, careful observation reveals that the percentage of the decrease is nearly the same for both Pentium© and ARM processors. However, the absolute values for the power consumption are significantly different, with the ARM processor configuration consuming more power. This is justified by the fact that ARM is a RISC processor. This means that in order to execute a statement in C, it must produce assembly code that is orders of magnitude bigger, comparing to the general purpose Pentium©. Thus, the ARM processor has to execute a very large number of total instructions, which finally translates to a large number of fetches from instruction memory. For this reason instruction memory power consumption is greater in the ARM processor than in the Pentium© processor. In Fig. 2 - 5, we can see the total power dissipated, which much greater for the ARM processor configuration comparing to the Pentium© processor configuration. This reveals, one more time, that instruction memory power dominates the total power budget, when programmable processors are used. Fig. 2 - 5 reveal that Hierarchical Search yields the best results concering power consumption, which combined with the low complexity and high performance SAD truncation distance criterion, we have the most optimized algorithm. in ME systems. MSE distance criterion, which has the highest computational complexity comparing with the other four, is responsible for the high power consumption in the algorithms that use it.

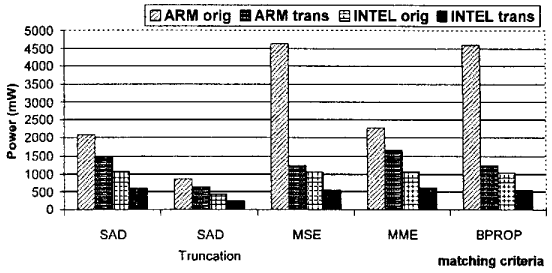

Fig. 2. Total Memory Power Consumption of the Full Search Algorithm

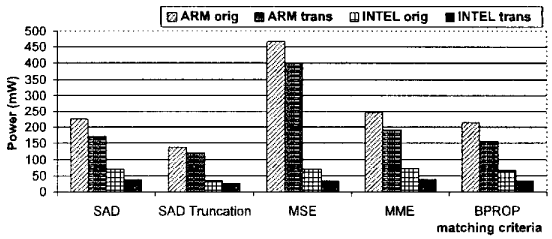

Fig. 3. Total Memory Power Consumption of the Hierarchical Search Algorithm

4.2. Performance

Generally speaking, most of the applied transformations have positive effect on the processor performance. However, the decrease is not significant. Typical values of the processor cycles are near 10% less after the transformations have been applied. However, we observed that the data-reuse transformation that produces power optimum solution does not give fastest implementation. Processor cycles for the original HS algorithm for two different architectures and a transformed version of it, are illustrated in Fig. 6. Results, concerning processor cycles for the other three algorithms are available but they are not presented due to lack of space. However, all of them, have nearly the same attribute concerning processor cycles. The great difference that is observed between the performance of the two different architectures, is due to the different operating frequency of each processor, where ARM operates at 30 MHz and Pentium© operates at 300 MHz.

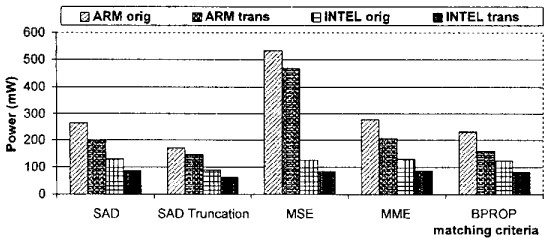

Fig. 4. Total Memory Power Consumption of the Three Step Logarithmic Algorithm

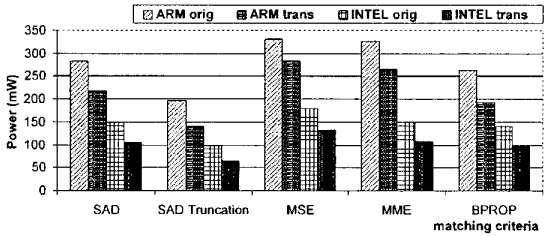

Fig. 5. Total Memory Power Consumption of the Parallel Hierarchical One-dimensional Search Algorithm

4.3. Memory Requirements

In Fig. 7 the effect of our methodology on memory requirements, is illustrated. From that it can be inferred that each transformation influences the necessary memory area in almost identical manner for all processor configurations. This is due to the fact that the use of each transformation implies the creation of a specific memory hierarchy. It is also clear that the transformations increase slightly the overall required memory, due to increased number of smaller memories. The fact that exist differences between different algorithms is justified, because every algorithm has different storing requirements.

5. CONCLUSIONS

Considering five matching criteria and two programmable processors, i.e. ARM and Pentium, a comparative study of four ME algorithms in terms of the performance, the energy dissipation, and the area, was presented. To perform our experimental results, we adopted a systematic optimization methodology. This methodology focuses both in memory power dissipation and system performance. Providing a plethora of alternative solutions/implementations of ME algorithms, a designer can easier lead to the most suitable for him application meeting certain design constraints.

6. REFERENCES

[1] A. P. Chandrakasan, R. W. Brodersen, *"Low Power Digital CMOS Design"*, Kluwer Academic Publishers 1998.

[2] D. Soudris, et. al., *"Data-Reuse and Parallel Embedded Architectures for Low-Power, Real-Time Multimedia Applications"*, Proc. of 10th Int. Workshop PATMOS 2000, Sep. 2000, pp. 243-254.

[3] S-C Cheng and H-M Hang, *"A Comparison of Block-Matching Algorithms Mapped to Systolic-Array Implementations"*, Trans. On Circuits and Systems for Video Technology, vol. 7, no. 5, October 1997, pp. 741-757.

[4] Y-K Cheng and S.Y. Kung, *"A Systolic Design Methodology with Applications to Full-Search Block-Matching Architectures"*, Journal of VLSI Signal Processing 19, 51-77, 1998.

[5] G. Gupta and C. Chakrabarti, *"Architectures for Hierarchical and other Block Matching Algorithms"*, Trans. On Circuits and Systems for Video Technology, 1995, pp. 477-489.

[6] P. Pirsch, N. Demassieux, and W. Gehrkw, *"VLSI architectures for video compression A survey"*, Proc. of IEEE, vol. 83, no. 2 pp. 220-246, Feb. 1995.

[7] T. Akari et al, *"Video DSP architecture for MPEG 2 codec"*, in Proc. of Int. Conf. On Acoustics, Speech, and Signal Processing (ICASSP), vol.2, 1994, pp. 417-420.

[8] F. Catthoor, et al., *"Custom Memory Management Methodology,"* Kluwer Academic Publishers, Boston, 1998.

[9] V. Bhaskaran and K. Kostantinides, *"Image and Video Compression Standards"*, Kluwer Academic Publishers, 1998.

[10] Peter Kuhn, *"Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation"*, Kluwer Academic Publishers, Boston 1999.

[11] P. Landman, *"Low power architectural design methodologies"*, Doctoral Dissertation, U.C. Berkeley, Aug 1994.

[12] J.M. Mulder, N.T. Quach, M.J. Flynn, *"An Area Model for On-Chip Memories and its Application"*, IEEE Journal of Solid-State Circuits, Vol. SC26,No.1,pp.98-105,Feb. 1991.

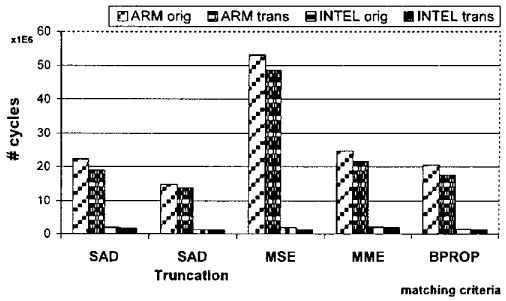

Fig. 6. Processor cycles of the HS Algorithm

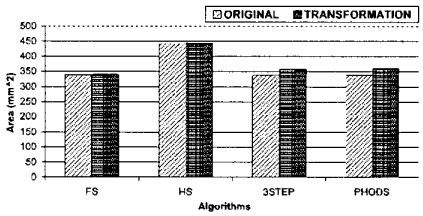

Fig. 7. Memory Requirements of the Motion Estimation Algorithms